REMARKS

The following remarks are responsive to the Final Office Action dated July 24, 2008. Applicant has amended claim 2. Claims 1–40 are pending.

Claim Amendments

Claim 2 has been amended for reasons unrelated to patentability. In particular, claim 2 has been amended to correct certain typographical errors. In claim 2, the word "wherein" was unintentionally spelled "wherin." This amendment has replaced "wherin" with "wherein." Applicant respectfully requests entrance of this amendment without the necessity of an RCE, due to the nature of this amendment.

Claim Rejection Under 35 U.S.C. § 103

In the Office Action, the Examiner rejected claims 1–2, 4–16, and 28–38, under 35 U.S.C. § 103(a) as being unpatentable over Kojima et al. (US 2003/0179705, hereinafter "Kojima") in view of Kaniz et al. (US 6,990,102, hereinafter "Kaniz"), and in further view of Hussain et al. (US 2003/0223361, hereinafter "Hussain"). The Examiner also rejected claim 3 under 35 U.S.C. § 103(a) as being unpatentable over Kojima in view of Kaniz and Hussain and in further view of Hirono et al. (US 6,910,213). The Examiner further rejected claims 21–28 and 39–40 under 35 U.S.C. § 103(a) as being unpatentable over Kojima in view of Kaniz. Applicant respectfully traverses the rejection. The applied references fail to disclose or suggest the inventions defined by Applicant's claims, and provide no teaching that would have provided a reason for modification thereof to arrive at the claimed invention.

"Heap" Per Applicant's Claims

To aid in the Examiner's understanding, Applicant notes that the term "heap" has several vastly different meanings in the field of computer science. In some contexts, "heap" can refer to a large pool of unallocated memory.[1] In these contexts, the term "heap" can be understood to mean "free memory." For example, Kaniz teaches two areas of Tables 601 and 602 in FIG. 6: a

[1] See, e.g., http://en.wikipedia.org/wiki/Dynamic_memory_allocation (describing dynamic allocation of memory: "[u]sually, memory is allocated from a large pool of unused memory area called the heap (also called the free store).") (parenthetical original)

"bin" area and a "heap" area. That is, addresses between 0 and n-1 in Tables 601 and 602 are referred to as "bin entries," whereas the remaining addresses (n to 4095) are referred to as "heap entries." Kaniz, col. 8, ll. 30–34. Kaniz differentiates between bin entries and heap entries only to the extent that bin entries are the primary destination of hash function output, whereas heap entries are the secondary destination of hash function output. That is, values are written into one of the heap entries only when a collision occurs after evaluating the hash function on a particular input value. Kaniz, col. 9, ll. 27–30. Therefore, the term "heap" in the context of Kaniz is best understood as this first use of the word "heap" in which the term refers to unallocated memory.

Hirono presents a similar understanding of the word heap. Hirono is directed to a memory management Hirono defines the "heap area" as the "work area of the memory used as needed while a program is being executed." Hirono, col. 2, ll. 13– 5. Thus, the heap area in Hirono is available as free memory for memory allocation, which is in accordance with this first understanding of the term "heap." *See, e.g.*, Hirono, col. 7, ll. 13–18. Neither Kojima nor Hussain provide any sort of definition for the term "heap," and neither Kojima nor Hussain use the term "heap" in their specifications.

None of the applied references use the term "heap" as it is used in the context of Applicant's claims or specification, however. Applicant's independent claims and specification define and use the term "heap" to refer to a specific tree-like data structure that in which the nodes of the tree conforms to a specific ordering. As explained in Applicant's previous response, this definition is consistent with a second usage of the term "heap" that is well-known in computer science.[2] Moreover, this definition and usage is quite different than a mere pool of unallocated memory, per the applied references. In this way, the Examiner has cited references that use the term "heap" in accordance with the first definition described above within the field of computer science, but these descriptions have little relevance to Applicant's claims

Specifically, claim 1 defines a heap as a tree that is in "heap-ordered representation." Claim 1 defines a heap-ordered representation as a representation in which "a value for the traffic statistics stored in any child node of the tree is less than or equal to a value for the traffic statistics stored within that child node's parent node within the tree." Thus, a heap in the

[2] See, e.g., http://en.wikipedia.org/wiki/Heap_(data_structure) (Describing a heap data structure as "a heap is a specialized tree-based data structure that satisfies the heap property")

context of claim 1 is a well-defined data structure having very particular properties, including the heap ordering feature as also clearly recited in claim 1. This understanding and use of the term "heap" is in harmony with Applicant's specification. *See, e.g.*, Applicant's specification, ¶ [0040]. None of the applied references understand or use the term "heap" as it is applied in the context of claim 1.

Applicant's Independent Claim 1

With this understanding in mind, Applicant's independent claim 1 requires maintaining a heap that provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows within the flow table. As stated above, claim 1 requires that the heap is a tree having a root node and a plurality of other nodes arranged as parent nodes and child nodes, each of the nodes storing traffic statistics for a different one of the packet flows. Claim 1 also requires that the tree is heap-ordered, in that a value for the traffic statistics stored in any child node of the tree is less than or equal to a value for the traffic statistics stored within that child node's parent node within the tree. The applied references, i.e. Kojima in view of Kaniz, Hussain, and Hirono, fail to teach, suggest, or disclose these requirements of Applicant's claim 1, alone or in combination.

Kojima

In the Final Office Action, the Examiner cited paragraph 56 and FIG. 4 of Kojima, as well as FIGS. 8 and 9 and paragraphs 61–64, in support of the rejection of claim 1. The Examiner asserted that these cited portions of Kojima teach the maintenance of a heap that provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows within the flow table. However, FIG. 4 of Kojima merely depicts a table 131, not a heap that provides a representation of a least a portion of the traffic statistics of the packet flows. Paragraph 56 of Kojima never mentions that table 131 is a heap. Instead, Kojima describes table 131 as being merely a simple table. The entire specification of Kojima lacks any reference to or disclosure of a "heap." To the extent that Kojima may discuss maintaining a table (i.e., table 131) representation of traffic statistics of packet flows, Kojima lacks any teaching whatsoever that the table is in any way related to a heap or structured in a manner that would anticipate or

suggest the requirements of claim 1. That is, Kojima fails to teach, suggest, or disclose the maintenance of a heap as required by Applicant's claim 1. Accordingly, Kojima fails to disclose this requirement of Applicant's claim 1.

The Final Office Action acknowledged that Kojima is silent with respect to a heap-ordered representation.[3] Given this to be true, it would be completely impossible for Kojima to disclose maintaining a heap that provides a representation of at least a portion of the traffic statistics of the packet flows as the term "heap" is used by the Applicant. A heap, by definition of Applicant's specification and the literal language of claim 1, requires a heap ordering. The Final Office Action therefore maintains two contradictory positions: that Kojima teaches maintaining a heap that provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows, but does not teach a heap-ordered representation. Particularly, the Final Office Action states both "Kojima teaches ... maintaining a heap (Fig. 4 element "traffic count table") that provides a heap-ordered representation (Paragraph 56)"[4] and "Kojima is silent in teaches [sic] providing heap-ordered representation."[5] As defined in Applicant's specification and as literally required by claim 1, a heap is a tree that is heap-ordered. That is, a heap must be, by definition, heap-ordered, so if Kojima fails to teach heap-ordering, Kojima necessarily fails to teach a heap that represents any sort of data, let alone traffic statistics of packet flows. Kojima fails to provide any such teaching, therefore Kojima fails to teach or disclose this requirement of claim 1.

Kaniz

Contrary to the Final Office Action's assertion, Kaniz fails to overcome this shortcoming of Kojima. Kaniz, like Kojima, fails to teach, suggest, or disclose maintaining a heap that provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows within the flow table as required by claim 1, so Kojima in view of Kaniz fails in this endeavor. Although Kaniz uses the term "heap," Kaniz defines a heap quite differently than the requirements of claim 1, as described above. Claim 1 requires that a tree is heap-ordered "in that

[3] Final Office Action dated July 24, 2008, p. 3 ("Kojima is silent in teaches [sic] providing heap-ordered representation.").
[4] *Id.* pp. 2–3.
[5] *Id.* p. 3.

a value for the traffic statistics stored in any child node of the tree is less than or equal to a value for the traffic statistics stored within that child node's parent node within the tree." Kaniz teaches no such ordering. Where Kaniz uses the term "heap entry," Kaniz is referring to an entry in table 2 (602, FIG. 6) between the addresses n and 4095. Kaniz, FIG. 6; col. 8, ll. 32–34 ("The remaining entries 604 are referred to as "heap entries" and have addresses from "n" to "4095."). Kaniz lacks any description whatsoever of Table 2 of FIG. 6 being a tree let alone a heap-ordered tree in the manner recited by claim 1.

In fact, Kaniz discusses an entirely different ordering of data that is not heap ordered. Kaniz states that the data is put in Tables 601 and 602 according to a hash function, rather than being maintained according to a heap. Kaniz, Abstract. Kaniz defines a hash function as a function that "generates an output value within a certain range based on an input value." Kaniz, col. 9, ll. 17–19. Kaniz describes the use of the hash function as a way in which to decide where among the bin entries to store a table entry. *See* Kaniz, col. 9, ll. 13–38. When a "collision" occurs (i.e., when the hash output selects the same bin entry twice, Kaniz col. 9, ll. 27–30), Kaniz states that the repeated table entry is stored in one of the heap entries, rather than the selected bin entry. Kaniz, col. 9, ll. 31–33. This description of a heap entry is entirely different than a heap-ordered representation as required by Applicant's claim 1. Therefore, Kojima in view of Kaniz fails to teach, suggest, or disclose maintaining a heap that provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows within the flow table as required by Applicant's claim 1.

Hussain

Hussain also fails to overcome this limitation of Kojima in view of Kaniz. Hussain lacks any explicit description of heaps or a heap ordering. The Final Office Action cited FIG. 3 as a tree having a root node and a plurality of other nodes arranged as parent nodes and child nodes, each of the nodes storing traffic statistics for a different one of the packet flows. Even if Hussain teaches the general notion of a tree structure, Hussain fails to overcome the limitations noted with respect to Kojima in view of Kaniz. Applicant's claim 1 requires that a heap be a heap-ordered tree. Claim 1 specifically requires heap ordering be "that a value for the traffic statistics

stored in any child node of the tree is less than or equal to a value for the traffic statistics stored within that child node's parent node within the tree."

Hussain teaches no such requirement or definition of heap ordering. Hussain fails to teach, suggest, or disclose that a **value** for traffic statistics is stored in any node of hierarchy 300 of Hussain. Instead, Hussain teaches that hierarchy 300 comprises a plurality of metering control blocks (MCBs). Hussain, ¶ [0046]. That is, rather than a being a heap that provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows, as required by claim 1, Hussain teaches that the MCBs receive the packet flows themselves. Hussain, ¶¶ [0046], [0048]. The MCBs must receive the packet flows in order to perform metering of the packet flows. FIG. 3 of Hussain shows the packet flows themselves entering each of the MCBs. This is incompatible with the requirement of Applicant's claim 1 that the heap be a heap-ordered representation of at least a portion of the traffic statistics of the packet flows within the flow table. Claim 1 requires calculating traffic statistics associated with packet flows through a network and storing the traffic statistics within a flow table. Hussain fails to disclose these requirements of claim 1, and there would be no way to modify the MCBs of Hussain to perform these requirements.

Furthermore, even if Hussain taught that one of the MCBs stored traffic statistics for packet flows through that particular MCB, Hussain lacks any teaching of a centralized location that would collect and/or store all of the information from each of the MCBs. That is, Hussain lacks any teaching of storing the traffic statistics within a flow table, as required by Applicant's claim 1. Claim 1 requires "a heap-ordered representation of at least a portion of the traffic statistics of the packet flows **within the flow table**." Thus even if each MCB were to collect traffic statistics for each of the packet flow associated with the MCB, Hussain still fails to disclose maintaining a heap that provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows within the flow table, because Hussain fails to teach storing the traffic statistics within a flow table.

Hussain teaches that the MCBs are devices or connected with individual devices, Hussain, ¶ [0048] ("first level MCBs 302 and 304 may meter packet flows from individual level devices"), which is contrary to the notion of a heap-ordered representation of a portion of a flow table, required by Applicant's claim 1. Thus even if the packet flows from lower level flows are

part of higher level flows, this does not necessarily imply a heap ordering, and even if it did (to which Applicant does not acquiesce), there would be no reason or obvious method by which to combine this teaching of Hussain with the teachings of Kojima and/or Kaniz.

To state this another way, let us assume, *arguendo*, that hierarchy 300 of Hussain does in some way guarantee a heap ordering (an interpretation of Hussain to which Applicant does not acquiesce). If hierarchy 300 were to, instead, be applied to a portion of traffic statistics of packet flows within a flow table, wherein the flow table stores traffic statistics, and wherein the traffic statistics are associated with packet flows through a network, **how would one of ordinary skill in the art learn a heap-ordered representation of at least a portion of the traffic statistics?** Hussain would merely teach that this occurs naturally when packet flows are filtered from a larger number of MCBs into a smaller number of MCBs. Hussain in no way teaches that this property is in any way useful or that this "ordering" is important. Moreover, Hussain fails to disclose how one would replicate this ordering outside of actually having the packet flows flow through the MCBs themselves. Thus, if one of ordinary skill in the art were to apply hierarchy 300 of Hussain to traffic statistics as opposed to the packet flows themselves, there would be absolutely no way to obtain the results required by Applicant's claim 1. This is because the heap ordering required by claim 1 does not occur naturally when collecting traffic statistics, as in the context of Applicant's claim 1.

Accordingly, it would not have been obvious to one of ordinary skill in the art to combine the teachings of Hussain with the other applied references. Moreover, even if one of ordinary skill in the art had combined the references, the combination still would not yield the requirements of Applicant's claim 1, for at least the reasons discussed above.

Hirono

Hirono also fails to overcome this inadequacy of Kojima in view of Kaniz and Hussain. Hirono defines the "heap area" as the "work area of the memory used as needed while a program is being executed." Hirono, col. 2, ll. 13– 5. This is contrary to the requirement of claim 1 that a heap be a tree, not a work area of memory. Moreover, claim 1 requires that the heap provide a heap-ordered representation of at least a portion of the traffic statistics of the packet flows within the flow table. Hirono fails to even teach that a heap is a tree structure, much less that a heap

provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows within the flow table. Accordingly, Kojima in view of Kaniz, Hussain, and Hirono fail to teach, suggest, or disclose the requirements of Applicant's claim 1.

None of the applied references disclose a "heap" or "heap ordering" in accordance with the requirements of Applicant's independent claim 1. To the extent the applied references discuss a "heap," the use of that term is quite different than that term is used and defined in claim 1. That is, to the extent the term "heap" is discussed in the applied references, that term refers to, generally, a pool of available memory, as discussed above. None of the applied references even attempt to define a "heap ordering." Moreover, none of the applied references, alone or in combination, disclose maintaining a heap that provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows. To the extent that the applied references may disclose traffic statistics, they fail to disclose that the traffic statistics are maintained in a heap that provides a heap-ordered representation thereof. Accordingly, the applied references fail to teach, suggest, or disclose maintaining a heap that provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows as required by Applicant's claim 1.

Applicant's Independent Claims 13, 21, 28, and 39

Independent claims 13, 21, 28, and 39 each comprise requirements similar to maintaining a heap that provides a heap-ordered representation of at least a portion of the traffic statistics of the packet flows, wherein the heap is a tree having a root node and a plurality of other nodes arranged as parent nodes and child nodes, each of the nodes storing traffic statistics for a different one of the packet flows, and wherein the tree is heap-ordered in that a value for the traffic statistics stored in any child node of the tree is less than or equal to a value for the traffic statistics stored within that child node's parent node within the tree. Therefore similar arguments as presented with respect to claim 1 apply with respect to these independent claims.

Dependent Claims

The dependent claims, i.e. claims 2–12, 14–16, 22–27, 29–38, and 40 incorporate the requirements of the respective independent claims. Accordingly, for at least the reasons discussed above, the applied references fail to teach, suggest, or disclose the requirements of the

dependent claims. Moreover, the dependent claims include a number of requirements likewise not taught or disclosed by the applied references.

For example, Applicant's claim 10 requires wherein maintaining a heap comprises performing a heapify operation on the heap after receiving the packet and updating the flow statistics. "Heapify" is an operation that maintains heap ordering of a heap. This definition is consistent with Applicant's specification at ¶ [0041], which states, "after updating the values associated with the nodes of the represented tree, control unit 22 executes a 'heapify' operation to ensure that each node of the represented tree has a value that is less than or equal to the value of its parents." Applicant's specification elaborates on the steps in performance of the heapify operation at ¶ [0058].

The Final Office Action asserted that Kaniz teaches a heapify operation at col. 8, ll. 30–39 and FIG. 6, referring to "heap entries." However, Kaniz does not disclose the term "heapify" in the cited portion. Moreover, Kaniz fails to disclose anything remotely similar to a heapify operation that would maintain a heap ordering in a heap structure, especially a heapify operation that is performed after updating flow statistics. Instead, Kaniz merely teaches the storage of data in accordance with a hash function. Storing data as a result of a hash function yields vastly different results than storing data in accordance with a heap data structure. For example, as opposed to a heap data structure, no sorting or ordering is guaranteed with respect to storing data in accordance with a hash function. Accordingly, Kaniz fails to disclose wherein maintaining a heap comprises performing a heapify operation on the heap after receiving the packet and updating the flow statistics as required by Applicant's claim 10. None of the other applied references teach this requirement, thus the applied references fail to disclose this requirement of Applicant's claim 10.

For at least these reasons, the Final Office Action has failed to establish a prima facie case for non-patentability of Applicant's claims 1–16 and 21–40 under 35 U.S.C. § 103(a). Applicant therefore respectfully requests withdrawal of this rejection.

CONCLUSION

All claims in this application are in condition for allowance. Applicant respectfully requests reconsideration and prompt allowance of all pending claims. Please charge any additional fees or credit any overpayment to deposit account number 50-1778. The Examiner is invited to telephone the below-signed attorney to discuss this application.

Date: September 16, 2008

By: Kent J. Sieffert

SHUMAKER & SIEFFERT, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

Name: Kent J. Sieffert
Reg. No.: 41,312